Exercise - 2 : Implementing E-Commerce Search Function

Linear Search :

1. It is one of the most basic searching techniques that simply traverses the given array of inputs one-by-one while checking them with the target.
2. Whenever an element matching the target is found, we return it’s details and stop the searching process.
3. In our program, we are searching for a product using a given target “product ID” in an array of products.
4. Whenever we find a product object whose “ID” attribute matches our target , we return that Object and stop searching.

Complexity Analysis :

Time Complexity :

1. Best case : O(1)

- It means constant time and it occurs when our target element is found in the very beginning of the array.

2. Worst Case : O(n)

- It means Linear time complexity where ’n’ is the size of given array.
- It occurs when our target element is found at the end of the list and when we have to check all the elements of the array.

3. Average Case : O(n)

- It occurs when our target is present anywhere between the first and last elements

Code :

```
package org.example;

public class LinearSearch {

    public Product search(Product[] prods,int id){

        System.out.println("searching for product with id "+id+" using linear search”);

        for(Product prod : prods){
            if( prod.prodId == id){

                return prod ;
            }
        }

        return null ;
    }
}
```

Binary Search :

1. It is a bit more advanced searching technique that is applied on sorted arrays.
2. Here, in each pass , we can reduce our search space half

3. We initialise 2 variables ‘high’ and ‘low’ where ‘high’ will be pointing to the last element in the array and low will be pointing to the last element in the array.
4. Then , we calculate ‘mid’ by using :

   mid = (low + high)/2
5. Now, we compare our target with the value at ‘mid’ index.
6. If arr[mid] == target : (i.e., ‘id’ of product at mid index is equal to the target id)

   return the product object at mid index
7. If arr[mid] < target :

   It means our target is present to the right of mid, so reduce our search space to the right half of the array by doing :

   low = mid+1
8. If arr[mid]>target :

   It means our target is present to the left of mid, so reduce our search space to the left half of the array by doing :

   high = mid-1
9. This procedure is repeated as long as “low<=high”:

   once low crosses high, we will come out of the loop and stop searching, it means we haven’t found our product, so we can return the appropriate message.

<u>Code :</u>

```
package org.example;

public class BinarySearch {
    public Product search(Product[] prods,int id){
        System.out.println("searching for product with id "+id+" using binary search");
        int l = 0;
        int r = prods.length - 1;
        int mid;
        while(l<=r){
            mid = (l+r)/2 ;
            if (prods[mid].prodId==id) {
                return prods[mid];
            }

            else if (prods[mid].prodId<id){
                r = mid-1;
            }

            else{
                l = mid+1;
            }
        }
        return null ;
    }
}
```

<u>‘Main’ class Code :</u>

```java
package org.example;
import java.lang.reflect.Array;
import java.util.*;

public class Main {
    public static void main(String[] args){
        Scanner sc  = new Scanner(System.in);
        Product[] prods = { new Product(51,"shampoo","dove"),
                new Product(101,"laptop","macbook"),
                new Product(27,"mobile","moto"),
                new Product(49,"tv","samsung"),
                new Product(36,"washing machine","whirlpool"),
                new Product(14,"book","clasmate")};

        System.out.println("Enter product id to search : ");
        int tar = sc.nextInt();

        //doing linear search
        LinearSearch linSearcher = new LinearSearch();
        Product lin_res = linSearcher.search(prods,tar);

        if (lin_res==null){
            System.out.println("Product not found through linear search");
        }
        else{
            System.out.println("Required product category : "+lin_res.category);
            System.out.println("Required product name :" + lin_res.prodName);
        }

        BinarySearch binSearcher = new BinarySearch();

        //doing binary search on sorted array
        Product[] sortedprods = Arrays.copyOf(prods,prods.length);
        Arrays.sort(sortedprods, Comparator.comparingInt(p -> p.prodId));
        Product bin_res = binSearcher.search(prods,tar);

        if (bin_res==null){
            System.out.println("Product not found through binary search");
        }
        else{
            System.out.println("Required product category : "+bin_res.category);
            System.out.println("Required product name :" + bin_res.prodName);
        }
    }
}
```

Product Class :

```
package org.example;

public class Product {
    int prodId;
    String category;
    String prodName;

    Product(int id,String cat,String name){
        prodId = id;
        category = cat;
        prodName = name;
    }
}
```